LEGIBILITY NOTICE

A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

LA-UR -88-2083

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

LA-UR--88-2083

DE88 014417

TITLE: A TRAINING PROGRAM FOR SCIENTIFIC SUPERCOMPUTING USERS

AUTHOR(S): Floyd Hanson, University of Illinois at Chicago
Thomas Moher, University of Illinois at Chicago
Nora Sabelli, University of Illinois at Chicago
Ann Solem, Group C-2

SUBMITTED TO: Supercomputing '88 Conference
Orlando, Florida
November 14-18, 1988

DISCLAIMER

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

A Training Program for Scientific Supercomputing Users

Floyd Hanson, Thomas Moher, Nora Sabelli, Ann Solem

University of Illinois at Chicago
Chicago, IL 60680-6998

Submitted to Supercomputing '88 Conference, November 14-18, 1988

Addresses:

Floyd Hanson. Mathematics, Statistics, and Computer Science, M/C 249, University of Illinois at Chicago, Chicago, IL 60680-6998, u12688@uicvm.bitnet, 312-996-8773.

Thomas Moher. Electrical Engineering and Computer Science, M/C 154, University of Illinois at Chicago, Chicago, IL 60680-6998, moher@uicbert.bitnet, 312-996-4562.

Nora Sabelli. Computer Center and Chemistry, M/C 135, University of Illinois at Chicago, Chicago, IL 60680-6998, u18214@uicvm.bitnet, 312-996-6806.

Ann Solem. Computer Center, M/C 135, University of Illinois at Chicago, Chicago, IL 60630-6998, u22210@uicvm.bitnet, 312-996-9138; on leave from Los Alamos National Laboratory, MS M995, Los Alamos, NM 87545, avs@lanl.gov, 505-667-5460.

Corresponding Author: Ann Solem
Presenting Author: Nora Sabelli
Presentation Media: Overhead viewgraphs.

ABSTRACT

There is need for a mechanism to transfer supercomputing technology into the hands of scientists and engineers in such a way that they will acquire a foundation of knowledge that will permit integration of supercomputing as a tool in their research. Most computing center training emphasizes computer-specific information about how to use a particular computer system; most academic programs teach concepts to computer scientists. Only a few brief courses and new programs are designed for computational scientists.

This paper describes an eleven-week training program aimed principally at graduate and postdoctoral students in computationally-intensive fields. The program is designed to balance the specificity of computing center courses, the abstractness of computer science courses, and the personal contact of traditional apprentice approaches. It is based on the experience of computer scientists and computational scientists, and consists of seminars and clinics given by many visiting and local faculty. It covers a variety of supercomputing concepts, issues, and practices related to architecture, operating systems, software design, numerical considerations, code optimization, graphics, communications, and networks. Its research component encourages understanding of scientific computing and supercomputer hardware issues. Flexibility in thinking about computing

needs is emphasized by the use of several different supercomputer architectures, such as the Cray X/MP48 at the National Center for Supercomputing Applications at University of Illinois at Urbana-Champaign, IBM 3090 600E/VF at the Cornell National Supercomputer Facility, and Alliant FX/8 at the Advanced Computing Research Facility at Argonne National Laboratory.

Keywords: Supercomputing, training program, remote supercomputing center, secondary supercomputing center, computational scientists, supercomputing support environment, supercomputing application environment, user experience.

INTRODUCTION

Increased Hardware Availability and Increased Supercomputing Education

In 1984 the National Science Foundation launched a program designed to bring supercomputing to scientists and engineers engaged in basic research outside the national laboratories [1], [2]. The availability of supercomputers in the Phase I and Phase II NSF-supported national centers, plus the NSFnet communications backbone, has had a profound impact on the way computational science is conducted and, perhaps more importantly, on the expectations of the scientific community.

In addition to improvements in hardware, software, and communications, supercomputing training is an important component of any national effort to advance the use of supercomputers. This importance of training was emphasized by the Panel on Large Scale Computing in Science and Engineering, chaired by Peter Lax in 1982 [1], and was restated in the report on a National Supercomputing Initiative, chaired by H. Reveche, D. Lawrie, and A. Despain in 1987 [3].

Increased capability in modelling physical systems requires advances in scientific software, as well as in hardware and systems. Advances in scientific application software have been mostly the result of increased access to mini- and minisuper-computers and are probably not yet the result of the increase in supercomputing access [4]. Full potential for development of sophisticated supercomputer software has yet to be realized. Scientific software for supercomputers lags behind minicomputing software for several reasons, among them: (1) lack of pervasive guaranteed access conducive to uninhibited exploration; (2) consequent lack, except in a few instances, of direct familiarity with the characteristics that transform a "computer" into a "supercomputer"; and (3) the lengthy time for development of large-scale software.

Where do we find supercomputing applications expertise? Supercomputing centers are typically headed by physical scientists engaged in computational science, rather than by computer scientists engaged in software or hardware development. So, it is to the potentially large pool of physical scientists and engineers, whose practice pushes them towards increasingly sophisticated use of supercomputers, that we must pay attention and address our efforts if effective supercomputer use is to follow closely on the heels of innovative supercomputer architecture design. Yet, this is a group that has traditionally learned computing by word of mouth within its own applied field; and the word-of-mouth approach is terribly archaic given the huge investments in computer hardware that have been made and the pace at which they will continue to be made. The National Supercomputing Initiative report [3] urges that "universities receive incentives to develop both undergraduate and graduate curricula for large-scale computing in engineering and science. The curricula must be interdisciplinary and foster a deeper understanding of the underlying problems in large scale computer modeling."

The task that we proposed for ourselves is more in line with technology transfer programs than with either supercomputing center training or academic computer science programs [3], [5]: to provide scientists at secondary supercomputing centers with enough background and experience to understand the concepts and issues of supercomputing as they relate to their own discipline and to integrate supercomputing as a tool in their research. Bringing professionals to the forefront of technology is nothing new: industry has engaged in it for a long time, and professionals in academia often use sabbatical leaves or their own research to fulfill the same need in an informal manner. What makes supercomputing unique now is the accelerated pace of supercomputer availability, the remote nature of supercomputer equipment (in most cases), and the lack of a sufficient body of experienced scientific programmers (outside of a few industrial and academic centers and the national laboratories, where supercomputer users are scientists who grew up with computers as computers were developed). Our program tries to provide an environment (utilizing remote access to supercomputers and providing comprehensive supercomputing support locally) where a group of experienced computational scientists can be built up much faster than it would grow otherwise.

Traditional computer science curricula do not, and cannot, provide the kinds of detailed background necessary to help non-computer scientists port their code to supercomputers. The typical computer science program rarely, if ever, offers a course in supercomputing, and such a course is more likely to be a generic course that considers a broad class of architectures than an in-depth investigation of programming techniques (for example, see [6]). Non-computer specialists may have little interest in the computer science aspects of these systems, and realistically, the knowledge they need may require several months of detailed study at the intersection of their application and the capabilities of the target machine.

Most supercomputing center training emphasizes computer-specific information about specific tools for specific computer systems. It does not come to grips with several crucial aspects of the scientist's needs: (1) problems arise after the scientist returns to home base, (2) many problems are due to networks and remote connections, and (3) the scientist often does not have the conceptual framework that is necessary in order to fully absorb the computing information in a few weeks. These problems are, of course, compounded at institutions without local supercomputer equipment. The very natural expectations created by the rise of "user friendly" microcomputer software tools and by the use of packaged software in disciplines without a long tradition of software development further complicate the problem.

Supercomputing Education for Non-Computer Scientists

That "supercomputing" is a topic of study on its own is not as obvious as it should be. An introduction to supercomputing certainly involves learning some principles of computer science, which short-term training programs cannot cover. Moreover, non-specialists typically do not have the time (or interest) to sit through a series of specialized computer science courses containing material that is of only selective relevance. On the other hand, the usual computer science survey courses have little to offer in this respect, since they are oriented toward a different audience. Furthermore, supercomputing imposes certain constraints (and opens up new avenues) that are not traditionally covered in computer science, so it is unclear whether traditional courses would really be all that helpful. For example, in the area of programming languages, it is one thing to rail against Fortran as an inappropriate vehicle for expressing algorithms, but if that is what you have, then the topic of interest is how to make best use of the tool you have, not what is the best tool [7].

Our approach is to capitalize on pressing research needs and on the traditional role of graduate and postdoctoral students as harbingers of new techniques for their research groups. To engage their full cooperation, the participant and his or her group must feel that they are accomplishing something of direct relevance to their research, must remain in constant touch with each other, and must come to understand the problems and challenges of supercomputing as they relate to their professional future. This is not to say that there is no need for scientists to spend long periods at supercomputing centers, nor that supercomputing center training workshops are not needed; rather, our program addresses the specific issues of incorporating supercomputing expertise in the shortest possible time into research groups not ready for other approaches and of making this expertise readily available locally.

METHOD

Goals

Today, computational scientists must be proficient in the use of a variety of supercomputers for their research. As national supercomputing centers become saturated, computer cycles will have to be used on whatever machines may be available. Developing technologies, such as parallelism, will likely worsen this situation for some time to come. Thus, the goals of our program are the following:

1. Prepare participants to become "educated consumers" of supercomputing technology.
2. Create adaptable people.
3. Bridge the gap between computer-specific training and computer science education.

Researchers need a training program that gives them the tools to adapt to multiple environments while focusing on their own research needs. We believe that participants must have significant time to view and re-view the same issues from different viewpoints, to learn more than simply how to use a particular supercomputer, and to learn more than abstract concepts. Direct presentations from different experts in a technical context and lack of emphasis on any single computer architecture broadens participants' views and helps prepare them to become "educated consumers" of this technology.

A difficult problem in developing supercomputer applications is the fine balance between portability across machines and optimization on a single machine. With such rapidly changing technology, it may be expedient to overtrain on a particular machine for a specific application, but this tends to subvert overriding educational goals. We are interested in training scientists and engineers about supercomputing concepts and issues, not about a particular supercomputer, because we know that over the course of their careers technology will change, change, and change again. We want people who can adapt to rapid technological changes and avoid becoming obsolete.

The result is a training program that attempts to bridge the gap between computer-specific training and computer science education. It treats scientific supercomputing conceptually and pragmatically. We have combined some features of both types of programs and added an emphasis appropriate for our audience [8]. This training program gains its strength from our long experience in education and the support of a traditional computing center combined with our more recent experience in the support of a variety of remote supercomputing centers and their users. It can be seen as a step in the evolution of an integrated network of centers of supercomputing. This training program does not teach participants how to use particular supercomputer systems, although they do learn this. It helps them understand what concepts and issues are central to

supercomputing. Thus, they will be better able to approach any supercomputer, evaluate its program development tools, adapt their application programs to the hardware, and evaluate the supercomputer's performance on application programs in their fields of research.

Computing Facilities: Critical to this robust approach has been access to not one, but several, advanced computing centers. Computers used in this program include: Cray X/MP48 running CTSS at the National Center for Supercomputing Applications (NCSA) at University of Illinois at Urbana-Champaign, IBM 3090 600E/VF running CMS at the Cornell National Supercomputer Facility (CNSF), IBM 3081 plus FPS 164 running ICAP at IBM Kingston, and Alliant FX/8 running UNIX at the Advanced Computing Research Facility (ACRF) at Argonne National Laboratory; and workstations include: IBM PC2/60, IBM PC/AT, Macintosh SE, and Sun.

Professional Level Audience: The program is designed for researchers in computationally-intensive research areas. Typically, the program has included graduate level science and engineering students who have completed their course work and are already engaged in computing as part of their research. The program is also appropriate for postdoctoral students and others who have recognized the need and who can devote a significant amount of time to research.

Prerequisite knowledge includes: (1) a working familiarity with Fortran programming and some mainframe operating system, and (2) a working application program. Because of the intensity of this workshop program, there is little time for significant code development, and participants are encouraged to have a working application program ready to explore, port, time, and debug. Appropriateness of the research code to supercomputers is a factor in selection of participants; if needed, we help students obtain code in their own field appropriate for the workshop. Participants work on the applications they know, and both students and advisors consider time spent in the training program to be part of their research experience.

Program Size and Time Commitments: The number of participants is kept small (four to six) to help develop a sense of "team work" that is emphasized by repeated participant presentations and by group debugging sessions with the staff. Participants focus on this training program and their own research full-time for eleven weeks without distractions from teaching or other duties. Student participants receive a research assistantship from their department or from the program so that they may devote their full time to the program. It is this immersion that allows participants to develop a better understanding about what makes computing a bottleneck and a rate-limiting step in their research.

Cross Fertilization of Research Areas: The training program is not limited to particular fields of research, but rather is open to any research area that needs the capabilities of supercomputers. Participants are selected from a balanced variety of disciplines so that they have the opportunity to learn from each other about a wider range of applications, problems, and solutions. This cross fertilization contributes to an understanding of the benefits to be gained from interdisciplinary interactions. Typically, the program has included two participants from applied fields and two from computer science, mathematics, or electrical engineering (see Table 1 and Table 5). It is our objective to maintain this balance for the foreseeable future; our experience with participants' interactions reinforces our expectations for the success of this approach.

All this is not to say that program participants do not need a background in computer science. The way to make the most significant improvements in the computing time of an application program lies more often in a better algorithm than in optimization or hardware improvements, and computer science knowledge often leads to these algorithm improvements. It is a secondary role of the computer science participants to help other participants gain an informal understanding of this value, while they themselves come to understand the application of supercomputers to scientific problem solving.

Table 1: Research Areas of Participants

Fall 1987 Program

Simulation of heat exchange and liquid propellants
Robotics and finite element stress analysis
Natural language interface for the development of large-scale software systems
Error-correcting, duadic cyclic codes

Winter 1988 Program

Time series modeling of electrical evoked potentials, as related to memory
Computational fluid mechanics
Image processing, analysis, and classification of diatoms
Parallel processing in sorting and searching of rule-based systems in artificial intelligence

Implementation

Rather than approach this training program from traditional points of view, such as hardware or software or algorithms or specific computers, we examined the nature of scientific supercomputing to reveal how it is different from other kinds of computing and, consequently, what knowledge and understanding is essential for computational scientists. As described by Lawson[9], many courses derive from the instructor's own background and educate people to have a similar outlook of relative importances. However, we felt that, having an application science orientation, a computational scientist should approach computing from neither hardware, software, nor algorithms, but from a blend and balance of all [10].

The program includes two components, seminars and research [11].

Seminar Component: Seminars cover many concepts and issues of current and future scientific supercomputing. These seminars are conducted roughly every other day by a variety of visiting lecturers--users, vendors, computer scientists, mathematicians, and engineers from academic departments, computing and research centers, and the computing industry--each an expert in his or her own field, and by students themselves and staff. The goal is to provide the breadth of topics that is necessary to understand scientific supercomputing in the practical environment of the participant's own application code running on a variety of supercomputers. Seminars emphasize general concepts of scientific supercomputing, supported by some specific technical details, so that participants are not completely overwhelmed and are able to retain enough of each topic to make sense out of the whole. We ask visitors to concentrate on similar concepts as related to their expertise so that participants are exposed to different points of view and learn, first hand, why there are many approaches to supercomputing and which ones suit their purposes better. Seminar topics are described below, and the relative weights of each can be seen in Table 2.

What is scientific supercomputing? To establish a foundation of knowledge, participants inventory their breadth of experience in computing: in hardware, in the use of existing software, and in the building of new software. We discuss characteristics that distinguish scientific supercomputing from other types of computing, such as student classroom computing, business comput-

ing, system programming, and real-time computing. This helps participants identify areas that are familiar and those that are new. The rest of the training program elaborates these distinguishing characteristics.

Architecture of supercomputers. We introduce principles, concepts, and terminology of supercomputer architectures; discuss vector processing hardware and multiple processors for parallel processing; and learn how several computers (such as Cray, IBM, and Alliant) work, including CPU, registers, and memory. Lecturers emphasize knowledge of hardware and system architectures and how they affect efficient coding techniques.

Operating systems and environments, communications and networks. We discuss concepts upon which different operating systems are based and examine several (such as CTSS, CMS, and UNIX) to understand important differences between them and how this affects scientific supercomputing. The importance of local area communications at the University and wide area communications via NSFnet and other national networks, including definitions and protocols, is included.

Software design and maintenance. Because scientific application programs have a long lifetime and are in a perpetual state of evolution as research needs change, we encourage special understanding of the need for good software design and analysis techniques and for re-evaluation of software after it has been implemented in a new environment. Modularity--both small-grain modularity for subroutine libraries and large-grain modularity for problem setup, computing, and post-processing--enhances readability and modifiability, two concerns that are dealt with in the research component of the training program.

Code optimization. Before a program can be made more efficient, it must be analyzed and timed to discover the most fruitful regions for optimization. The effect of choice of data structures on execution time, space/time trade-offs, and the need to discover better algorithms are discussed, as is automatic compiler optimization and the effects of vectorization and parallelization on coding and algorithms.

Numerical considerations. Computational scientists need to learn about the nature of computer arithmetic, round-off errors, random number generators, the stability of algorithms, and the different nature of errors in vector and parallel processing. Evaluating and selecting algorithms from existing libraries is another topic that does not have a regular place in curricula and that is of practical importance to our audience.

Computer graphics. Because of limited graphics hardware availability and current performance problems of graphics in a networked environment, some participants have little experience with computer graphics. These sessions expose participants to a wide variety of graphics, set their future sights on the importance of graphical output in the analysis of results, and help them understand the use of graphics as a tool in thinking about science. It also helps them understand the difference between lower-cost scientific graphics for day-to-day research and higher-cost presentation graphics for final research results.

Evaluation of program design and implementation. We emphasize the need to assess a computer program as a separate step: to remember the scientific problem we are exploring, and to evaluate if this program is the best mapping of the problem onto the computer and if this computer is the best choice for this problem.

Table 2: Seminar Topics

Topic	Number of Sessions
What is Scientific Supercomputing?	1
Introduction to Supercomputer Architectures and Operating Systems	2
Supercomputer Architectures: Cray, IBM, Alliant, experimental	3-5
Operating Systems and Environments: Cray, IBM, UNIX	3-5
Communications and Networks	2
Software Design and Maintenance	3
Code Optimization:	
Timing and Debugging	1
Vectorization	2-3
Parallelization	2-3
Numerical Considerations	2
Computer Graphics	2
Evaluation of Design and Implementation	2
Site Visits: NCSA, Argonne, other	2-4
Presentations:	
Students	3-5
Advisors	1-2
Visitors	1
Wrap-Up	1

Research Component: Each participant defines, early in the program, and then pursues his or her own workshop research objectives by implementing an application program on a variety of supercomputers and settling on particular criteria for understanding. Participants present aspects of their research several times during the program as their application advances, and receive suggestions from staff and other participants on the methods used. In this way, the team continues learning from each other's work. Advisors' presentations are an important part of this process and help emphasize the relationship of the workshop to the participant's career.

After adapting their codes to particular supercomputer architectures and operating systems, participants spend several days at a supercomputing/research center to work closely with researchers and experts in studying their application programs and the effects of supercomputing environments. The site visit provides participants with intensive experience working on their application programs with the close assistance of on-site experts.

To facilitate the participants' research, "toolkit" and "clinic" sessions are conducted by workshop staff and faculty sponsors; these are a more formal learning part of the research (or laboratory) component of the workshop.

Half-hour toolkit sessions assist participants in identifying useful computing tools and related documentation for particular computers. We want to emphasize that each toolkit covers a variety of operating systems and encourages a system-independent comparative attitude. Topics covered in toolkit sessions are given in Table 3.

Table 3: Toolkit Topics

Topic	Number of Sessions
Signing On and Off, and Shipping Files	1
Finding Information	1
Compiling, Loading, and Executing, and Their Listings	1
Editors	3
Debugging Tools	2
Timing and Analysis Tools	3
Vectorization	2
Parallelization	2
Controllers and Batch Processing	3
Building User Libraries	1
Math Libraries	1
Computer Graphics	2

Clinics provide an opportunity to discuss particular programming problems and issues in detail with experts and students together; these team discussion/debugging meetings cover students' actual problems. For example, we have guided tours of how to read and debug programs, including those written by others, a very important part of all our professional lives. Other successful clinics include design of testing procedures for each code and participants' presentations of several different approaches to a problem gathered from the literature. These clinics, together with student and staff presentations, serve a fundamental purpose: they provide a computing "apprenticeship" that is the counterpart of the apprenticeship provided by students' advisors in their scientific field. We have found, time and time again, that these sessions are where participants learn the limitations of their own approaches and start to develop wider perspectives. Topics covered in clinics are given in Table 4.

Facilities and Support: Each participant has exclusive use of a workstation connected to local and national networks; the proximity of participants throughout the program contributes to interchange of ideas and understanding. Participants also have easy access to local and remote consultants, online information systems, and a local library of materials, which includes books, journals, supercomputer manuals, articles on computer optimization, mathematical libraries, videotapes, and online pointers to this material. Into the existing online computer documentation system, we have incorporated an online bibliography of materials available in our local library and in the University library. This integration of mainframe and micro-computer documentation with the supercomputing bibliography has two objectives: (1) it makes clear to the student that supercomputing is not a cure-all but must be used only when it is effective, and (2) it introduces, to the rest of the computing community, the idea that supercomputing can be used to expand local computing capabilities.

Table 4: Clinic Topics

Topic	Number of Sessions
Compiler and Loader Listings	1
Reading and Reviewing Programs	1
Testing	2
Timing and Debugging	1
Vectorization	1
Parallelization	1
Reading the Literature	1
Computer Graphics	1
Optimization and Comparing Supercomputers	1

RESULTS

When the program was designed, we recognized that its unique features should be continuously evaluated, since there was no previous experience to draw upon. The most useful feedback is that from the students themselves; we distributed detailed evaluation sheets after each seminar, toolkit, and clinic session; these sheets were reviewed by the program staff weekly. We also scheduled a post-mortem session with each group of students. The most important conclusions of this evaluation process were the following:

1. The "immersion workshop" approach was extremely successful in beginning to create robust, flexible supercomputer users. Immersion led to a qualitative change in the students' understanding of the process of computing and its relationship to their scientific aims.
2. Research assistantships give participants the time and support to immerse themselves in this program.
3. Access to different computer architectures and to supercomputing centers of expertise is fundamental; the program would not exist without them.
4. Direct exposure to vendors in a technical context was an important part of the learning experience; this analytical interaction with vendor experts is characteristic of the forefront of technology.
5. Repetition of presentations by the students as their work progressed was very useful to the students themselves and an effective method of sharing strategies.
6. We should strive for greater participation of students' research advisors.
7. Site visits are important to the sense of growth, understanding of research computing trends, and camaraderie among participants.
8. Perhaps the most important and surprising comment was that every student learned basic computing and debugging techniques during the program that would have been useful in software development before the program.

A good way to show the success of the program is through its effects on participants and their research groups. Because of their new understanding of fundamental concepts and their newly perceived need for information, two engineering alumni will take computer science classes next fall

in numerical algorithms and introduction to computer architecture. An alumnus was able to demonstrate, by careful benchmarking of a speeded up program, an error that cropped up after 4000 iterations and opened the way for making the group results more accurate. We have a second applicant from this same research group, which is already a heavy user of supercomputers at CNSF and Palo Alto IBM 3090 installations. An alumna who was not computer proficient before the program was able to expand the complexity of her thesis research in one quarter; and we have a second applicant from the same field. An engineering alumnus was asked by his department head to make a presentation to the whole department on his timing benchmarks on finite element program optimization; we have had an applicant from that department every time the program has been offered.

The main results of the program are discussed below.

Workshop approach success: Immersion of a few students into concepts, issues, and practices of scientific supercomputing for a relatively long time was fundamental. Participants learned a great deal about supercomputing in general, adapted their application programs for several supercomputers, and compared performance. We have started to see participant research develop with a better understanding of the nature of supercomputing; we can also see that links are by no means cut at the end of one term and alumni draw freely on our resources as needed.

Changes in point of view: Students who have participated in our model program have learned how to diminish the role of computing as a bottleneck in their research. They are able to assess the value of different computers; as one participant put it, "I am more relaxed at the terminal. I also am able to implement new programs more quickly than before and I 'dream bigger', i.e., consider implementing programs that I would have considered to be too difficult before." We have also seen an increase in interdisciplinary questions among participants themselves and among participants and faculty engaged in the clinic sessions.

Direct application on own codes: Perhaps the single most important factor in the success of the program is the requirement that participants bring to the training program a specific problem that they need to solve. The danger of such an approach is that participants may over-specialize on their specific applications; in fact, however, our experience has been quite the opposite. By providing a firm focus for their work, participants have a much stronger motivation for assimilating new information, and they treat the workshop as much more than just an intellectual exercise.

Education of non-programmers: Of wider implication is the realization that there is no mechanism in place for non-programmers to learn "good programming habits," since these are still passed on in an apprenticeship context and professional scientific programmers are not common in academia. Additional avenues are needed for non-programmers to gain the background that is essential to their development and productivity.

Efficiency of the educational process and resource allocation: In evaluation sessions with current and past students, it has become obvious that there are two things students perceive as most useful: (1) the time available to go over and over again their own programs, learning something new as time and their background progress, and (2) sitting down with experts to deal with specific issues--be it what they need to know in order to move from fully supported mainframe use to almost unsupported supercomputer use, or how they time and speed up difficult code. Participants have been very clear that they consider the program a success from their standpoint: they have advanced in their research at the same time that they have absorbed an enormous amount of background education. The fact that this knowledge makes participants more valuable to their present research groups and more employable in the future plays no small part in their dedication.

This program has been able to bring to campus professionals not otherwise accessible. Visitors come as seminar speakers, but the program has waiting for them, in addition to the usual interested audience, a small group of dedicated students with the time and knowledge to get more out of the visit. The program has increased communications between the graduate student cohort on campus and computing center staff in charge of research computing support; it has allowed us, as computer professionals, to pass on to participants much more than what we were able to before. Staff whose time commitment is substantial consists of one full-time equivalent person during the two initial quarters of program design, redefinition, and search for support; this will be reduced to three-quarters of a person for the third quarter. But, we should not lose sight of the fact that a great part of this effort (for example, user contact and keeping abreast of developments in the field) would be expended by center staff in any case; in this sense, the program has provided an efficient and satisfying framework for furthering support of computer teaching and research on campus.

Unanticipated findings: The best lecturers for this training program are experts who understand the concepts and issues and can interpret the significance of various practices. People with limited skill in teaching or narrow experience in the use of supercomputers were not suitable.

Although we give invited speakers an outline of the whole training program, including the seminars they are involved in, we cannot predict what will actually be covered in each seminar. Thus, it is important that the workshop coordinator attend all seminars, toolkit, and clinic sessions to maintain knowledge of what has been covered, to identify topics that need further elaboration, and to help participants relate material to what has been covered and what will be covered.

Technical presentations on computer architecture and program optimization by computing industry experts are invaluable for their thorough knowledge of their own machines. It exposes participants to a breadth of viewpoints that increases their analytical skills.

We were surprised how important site visits were. Site visits contributed significantly to the development of shared experience of participants from different research areas. They afforded an opportunity for participants to break from their absorption of the large quantity of new material and spend some less pressured time with each other. As participants acquired common experiences, their discussions and interactions grew.

Students' advisors are a valuable resource of knowledge and experience from which participants can benefit. Involvement of advisors also increased their own understanding of the role of the program and its significance for their students. This involvement must be encouraged; it does not necessarily come by itself.

Participants did not have realistic expectations; they did not anticipate how beneficial this program would actually be or how much they would want to accomplish during it. Thus, we emphasize the value of starting their research component as soon as possible.

FUTURE DIRECTIONS

Program Content: The intellectual content of a training program of this nature is determined by its participants' needs; adaptability is one of its advantages. It is easy to imagine that in several years many topics covered now (vectorization and portability, for example) may have become part of the arsenal of tools of most computer-sophisticated research areas and will demand supercomputing center consulting rather than background educational programs. If our approach is successful, though, we will by then have evolved toward both the new needs in the current constituency (*more parallelism, more visualization, and more workstation/supercomputer interface* development) that are barely touched now and the new needs of an expanding new constituency, those social and medical scientists and humanists that now shy away from supercomputing. To fulfill the hopes for supercomputing, we have to seek these non-traditional users, adapt the program content to their research needs, and adjust the program size to accommodate them. In the same way that mathematicians and computer scientists in the program are important resources for the applied scientists in the team, we will use the expertise of applied scientists more comfortable with computing issues to expand the computing vocabulary of newer groups.

Adaptation for Other Scientific Computing Centers: The environment that we create for participants is, in terms of hardware and connectivity, the same one provided for their research by the University. In this sense, the program can be easily duplicated by any institution involved in computing support; in particular, any university affiliated with an NSF-supported supercomputing center is probably in a good position to duplicate the program with limited local effort. Discussions with colleagues at national laboratories (Argonne and Los Alamos, in particular) and with industry have made evident the appeal of this approach to other institutions engaged in transferring professional knowledge. As said in the introduction, this program owes much to an understanding of the way in which professionals keep abreast of new developments in technical fields.

For others seeking to emulate what we have done, we emphasize the importance of a broad-based pool of human resources and, particularly, communication and cooperation among all academic units that might be affected. Certainly, participation from computer science, mathematics, engineering, physical sciences, and computing centers is essential; ideally, social scientists, artists, and others would help round out the program and open the eyes of their colleagues to potential applications of supercomputers. Given the emphasis we place on team work and interactions, it is not enough just to advertise the training program. For the program to be successful, it is necessary to search for appropriate workshop participants and instructor expertise. This is another by-product of a multi-disciplinary base, in that the more disciplines that are actively involved in the program, the broader the pool of potential participants. It is also important to remember that, in a fast moving field like this, no single entity can count on in-house expertise necessary to realize the full potential of this program. Vendors, supercomputing centers, national laboratories, and academia must be part of the human resource pool to be drawn upon. Technical presentations by vendors and students' own advisors are a rich source of information not otherwise available; the potential for expansion has been barely tapped.

SUMMARY

The UIC Workshop Program in Scientific Supercomputing is presented as a link between supercomputing centers and their users and as a link between computational scientists and computer scientists. As such, it cannot exist on its own but must be coordinated with all of these. Success with the program allows us to offer it with confidence as a viable program for other universities with limited local supercomputing support but with a strong computing research environment. Its dissemination benefits not only individual universities and their supercomputer users; it also benefits national supercomputing centers as well, by providing these centers with a larger pool of supercomputer-sophisticated users and relieving their consulting staff of some onerous and repetitive questions. It increases interaction between computer scientists and the scientists who benefit from their research and who are often unaware of how effective a tool computer science can be. Also, we should not forget that, in the last instance, integration of supercomputing into everyday science and engineering practice depends on a large pool of trained users; development of new algorithms and techniques depends on a large pool of sophisticated users. These users are being formed at universities, and the rate at which supercomputing expertise diffuses through a university has a fundamental importance in the formation of the next pool from which we all will draw.

ACKNOWLEDGMENTS

This workshop program was made possible through the generous support and contributions of many people. Joint sponsorship within the University (UIC) provided a breadth of support that gave this program necessary validity. George Yanos, from the Computer Center, UIC, contributed to the program development; Steve Roy, from the Supercomputing Support Office of the Computer Center, UIC, provided consulting support; and Maxine Brown, from the Electronic Visualization Laboratory, UIC, provided graphics coordination. Seed money from the University's Graduate College was essential in funding participants, and research assistantships from the Departments of Chemical Engineering and Mechanical Engineering demonstrated their confidence in this program. Office space, seminar space, staff from the Computer Center were essential.

This workshop program could not exist without the technological base, supercomputing support, and computer time provided by the National Center for Supercomputing Applications, the Cornell National Supercomputer Facility, and the Advanced Computing Research Facility at Argonne National Laboratory. Support of the computing industry was vital in providing expert visiting lecturers and workstation equipment. We thank Kelly Altman of Cray Research and Kevin Quinn of IBM for their support in obtaining excellent speakers. And we thank IBM for their generous loan of equipment for participants and for supporting a Vectorization Fellowship for a student to act as class assistant and consultant. Site visits significantly affected the viewpoint of the participants and contributed to their growing understanding of scientific supercomputing. We thank Enrico Clementi and Douglas Logan of IBM Kingston, James Bottum and Alan Craig of NCSA, and Rick Stevens of Argonne National Laboratory for supporting and arranging visits to their sites.

Students made a significant commitment of their time based only on our promise to provide a worthwhile program for them; their names are listed in Table 5. And their advisors supported the students in this endeavor. And perhaps most importantly, we thank the many visiting lecturers for their time and effort in creating seminars of value to our students and for their long, fruitful discussions with the workshop staff; their names are listed in Table 6.

Table 6: Faculty

Associated Faculty

Kathy Barbieri - Cornell National Supercomputer Facility
Ingrid Bucher - Los Alamos National Laboratory
Duncan Buell - Supercomputing Research Center, Institute for Defense Analyses
Helen Doerr - Cornell National Supercomputer Facility
Evelyn Goldfield - Cornell National Supercomputer Facility
Robert Haber - National Center for Supercomputing Applications
John Lacher - Cray Research
Rick Lawrence - IBM Kingston
David Levine - Argonne National Laboratory
Douglas Logan - IBM Kingston
Michael Norman - National Center for Supercomputing Applications
David Soll - IBM Kingston
Craig Upson - National Center for Supercomputing Applications
Tony Warnock - Cray Research at Los Alamos National Laboratory

UIC Faculty

Maxine Brown - Electrical Engineering and Computer Science, UIC
Earl Gose - Bioengineering, UIC
Floyd Hanson - Mathematics and Computer Science Division, Argonne National Laboratory; and Mathematics, Statistics, and Computer Science, UIC
Tom Moher - Electrical Engineering and Computer Science, UIC
Sohail Murad - Chemical Engineering, UIC
Nora Sabelli - Computer Center, UIC; and Chemistry, UIC
Shin- Min Song - Mechanical Engineering, UIC
Jose Sousa - Electrical Engineering and Computer Science, UIC; and Borg Warner
George Yanos - Computer Center, UIC; and Mathematics, Statistics, and Computer Science, UIC

John Andrews - Computer Center, UIC
Peter Asick - Computer Center, UIC
Fred Damen - Computer Center, UIC
Steve Roy - Computer Center, UIC
Ann Solem - Computer Center, UIC; and Computing and Communications Division, Los Alamos National Laboratory
Michael Sperberg- McQueen - Computer Center, UIC
Kathi Suchy - IBM Vectorization Fellowship

Table 5: Student Participants

Fall 1987 Program

Victor Dirda - Electrical Engineering and Computer Science, UIC
Vanessa Job - Mathematics, Statistics, and Computer Science, UIC
Prasad Ravi - Chemical Engineering, UIC
Nazeer Shareef - Mechanical Engineering, UIC

Winter 1988 Program

James Choi - Electrical Engineering and Computer Science, UIC; and Bioengineering, UIC
Matt Hettinger - Physiology and Biophysics, UIC
Anil Manhapra - Mechanical Engineering, UIC
Li Zhou - Mathematics, Statistics, and Computer Science, UIC

REFERENCES

1. *Report of the Panel on Large Scale Computing in Science and Engineering*. Peter Lax, Chairman. Sponsored by the U.S. Department of Defense and the National Science Foundation, in cooperation with the Department of Energy and the National Aeronautics and Space Administration, Washington, D.C., December 26, 1982.

2. *A National Computing Environment for Academic Research*. Marcel Bardon and Kent Curtis, NSF Working Group on Computers for Research, National Science Foundation, July 1983.

3. *A National Computing Initiative: The Agenda for Leadership*. Chaired by Harold J. Raveche, Duncan H. Lawrie, and Alvin M. Despain. SIAM Workshop, Leesburg, Virginia, February 2-3, 1987. Society for Industrial and Applied Mathematics, Philadelphia, 1987.

4. Saxe, Paul, Los Alamos National Laboratory. Unpublished report, presented to *Theory and Simulation in Chemistry: The Impact of Minisupercomputers and Supercomputers*, organized by Thom H. Dunning, Jr. and Ron L. Shepard, Argonne National Laboratory, and Lola Anacker, John von Neuman Center, 1987.

5. Hyman, James M. "Future Directions of Large Scale Scientific Computing," in *Large Scale Scientific Computation*, edited by Seymour V. Parter, Academic Press, 1984, pp. 51-83.

6. Berztiss, Alfs. "A Mathematically Focused Curriculum for Computer Science," *Communications of the ACM*. Vol 30, No 5, May 1987, pp. 356-365.

7. Knuth, Donald E. "An Empirical Study of FORTRAN Programs," *Software--Practice and Experience*, Vol 1, 1971, pp. 105-133.

8. NCSA Summer Training Seminar course notes. National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, June 1986.

CNSF Smart Node training course notes. Cornell National Supercomputer Facility, Cornell University.

Anderson, Jess. "Local course near leading edge," *MACC News*. Academic Computer Center of the University of Wisconsin-Madison, Vol 22, No 7, August 24, 1987, pp. 4-5.

Hanson, Floyd B. Numerical methods for parallel and vector computers syllabus. Department of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago.

Maisel, Merry. "Educating Rita," *Gather/Scatter*. San Diego Supercomputer Center, Vol 3, No 7, July 1987, pp. 1-3.

Moses, Greg. Private communication. University of Wisconsin, August 1987.

Moyer, Tom. Private communication. Department of Civil, Mechanical, and Environmental Engineering, George Washington University, August 1987.

Fanoff, Robert M. Private communication, computational physics course notes, and NCSA Summer Institute course notes. Department of Physics, Kansas State University, August 1987.

9. Lawson, Harold W., Jr. "Computer Architecture Education," in *New Computer Architectures*. Edited by J. Tiberghien, Academic Press, 1984, pp. 225-286.

10. *UIC Workshop Program on Scientific Supercomputing, Fall 1987 Quarter*. University of Illinois at Chicago, 1987.

UIC Workshop Program on Scientific Supercomputing, Winter 1988 Quarter. University of Illinois at Chicago, 1988.

11. Dwyer, Thomas. "Discussant Remarks," *Computer Literacy: Issues and Directions for 1985*, National Goals for Computer Literacy in 1985 Conference, Reston, Virginia, December 18-20, 1980. Edited by Robert J. Seidel, Ronald E. Anderson, and Beverly Hunter, Academic Press, 1982, pp. 193.